Attorney Docket No. 30014200-1027

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of	)	Group Art Unit 2194
	)	
Hiura *et al.*	)	Examiner: Phuong N. Hoang
	)	
Application No. 09/488,909	)	
	)	
Filed: January 21, 2000	)	
	)	
For: METHOD FOR ENABLING MULTIPLE	)	
CONCURRENT SUBPROCESS	)	
HANDLING ON A SYSTEM USING A	)	
GLOBAL PROCESS	)	

Mail Stop Appeal Brief-Patents
Commissioner for Patents
Alexandria, VA 22313-1450

TRANSMITTAL OF APPELLANTS' BRIEF ON APPEAL

Dear Sir:

Appellants submit, in triplicate, Appellants' Brief on Appeal under 37 C.F.R. § 1.192 in support of the Notice of Appeal filed on October 18, 2005. Appellants also submit a check in the amount of $500 for the appeal brief fee as required by 37 C.F.R. § 41.20(b)(2).

The Commissioner is hereby authorized to credit overpayments or to charge any deficiency in a required fee to Deposit Account No. 19-3140. A duplicate copy of this sheet is enclosed.

Respectfully submitted,
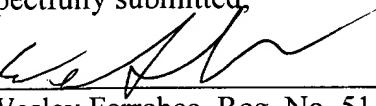
Dated: December 28, 2005   By: _____
A. Wesley Ferrebee, Reg. No. 51,312

Customer No. 58328 (formerly 26263)
SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box 061080
8000 Sears Tower
Chicago, IL 60606-6404
Telephone: (202) 408-6832
Facsimile: (312) 876-7457

Attorney Docket No. 30014200-1027

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of)	Group Art Unit 2194
)	
Hiura *et al.*)	Examiner: Phuong N. Hoang
)	
Application No. 09/488,909)	
)	
Filed: January 21, 2000)	
)	
For: METHOD FOR ENABLING MULTIPLE)	
CONCURRENT SUBPROCESS)	
HANDLING ON A SYSTEM USING A)	
GLOBAL PROCESS)	

Mail Stop Appeal Brief-Patents
Commissioner for Patents
Alexandria, VA 22313-1450

APPELLANTS' BRIEF ON APPEAL

Dear Sir:

In accordance with the provisions of 37 C.F.R. § 1.192, Appellants submit this Brief in

support of the Appeal for the above-referenced application.

I.   REAL PARTY IN INTEREST

The real party in interest in the present appeal is the Assignee, Sun Microsystems, a U.S.

corporation. The Assignment was recorded in the U.S. Patent and Trademark Office.

II.   RELATED APPEALS AND INTERFERENCES

There are no related appeals and no related interferences.

III.   STATUS OF CLAIMS

Claims 1-21 are pending in the above-identified application. Claims 1-21 were rejected in the Final Office Action dated April 19, 2005, and are now appealed. A listing of the claims appears in Appendix A.

IV.   STATUS OF AMENDMENTS

A Request For Reconsideration After Final was filed on August 19, 2005, wherein no amendments of the claims were made. An Advisory Action was mailed September 13, 2005, stating that the Request For Reconsideration After Final had been considered but rejected because the Request did not place the application in condition for allowance.

V.   SUMMARY OF INVENTION

Various embodiments of the invention are directed to means and methods for providing an interface to a master process and kernel of an operating system in a computer system. User-specific processes are mapped to virtual address spaces that overlay those of the master process. Thus, when a user-specific instruction is encountered by the master process, execution is transferred to the appropriate user-specific process. Because the user-specific process and the master process share a virtual address space, data can between transferred between the two processes with serialization, thus reducing processing overhead.

Claims 1, 8 and 15 are directed to a method, computer-readable medium, and apparatus for providing for concurrent subprocessing of a master process. By way of example and not limitation, the means for performing this method may be a virtual memory separator (identified

by numeral 232 of Figure 2) that is part of an operating system. The virtual memory separator 232 interfaces with a master process when a user-specific operation is encountered. (*See* master process 302 and user-specific operation 310 of Figure 3; steps 410, 412, and 414 of Figure 4; and pages 5, line 20 to page 7, line 22 of the Specification). The virtual memory separator 232 facilitates mapping one ore more user-specific processes so that they overlay or are equivalent to the virtual addresses of the master process (*e.g.*, user specific processes 304, 306, and 308 are mapped to master process 302). (*See* Figure 3 and page 5, line 20 to page 6, line 18 of the Specification). The virtual memory separator 232 further provides for processing the user-specific operation in the user-specific process. (*See* steps 410, 412, and 414 of Figure 4 and page 6, line 9 to page 7, line 22 of the Specification).

Claim 12 is directed to a computer system for enabling concurrent multiple subprocess handling in a global process environment. The computer system includes a global process (*e.g.* master process 302, and a virtual memory separator (*e.g.* virtual memory separator 232) that maps a user-specific process (*e.g.* locale-dependent process 304) to virtual addresses that mirror virtual addresses of the global process. (*See* page 5, line 20 to page 7, line 22 of the Specification and Figures 3-4). The user-specific process having an interface that mirrors an interface of the global process. (*See* page 6, lines 9-18 of the Specification).

VI. ISSUES

The issues to be reviewed are whether claims 1-11 and 15-21 are properly rejected under 35 U.S.C. § 103(a) as unpatentable over *Hetherington et al.* (U.S. Patent No. 6,275,810, hereinafter "*Hetherington*") in view of *Kaufman* (U.S. Patent No. 5,313,647), and whether claims 12-14 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over *Kaufman* in view of *Hetherington.*

VII.   <u>GROUPING OF CLAIMS</u>

The claims do not stand or fall together. Specific arguments as to the separate patentability of selected claims have been presented.

VIII.   <u>ARGUMENT</u>

To render a claim unpatenable under 35 U.S.C. § 103(a), the Examiner bears the burden of establishing a *prima facie* case of obviousness. *In re Rinehart*, 531 F.2d 1048, 189 USPQ 143 (CCPA 1976); *In re Linter*, 458 F.2d 1013, 173 USPQ 560 (CCPA 1972). In order to establish a *prima facie* case of obviousness, all of the claim limitations must be taught or suggested by the prior art. *See* MPEP 2143.03, *citing In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). In the present application, the cited prior art fails to teach or suggest all of the limitations of any of pending claims 1-21. Thus, the Examiner has failed to establised a *prima facie* case of obviousness for any of pending claims 1-21. Accordingly, the rejection of claims 1-21 should be reversed.

A. Claims 1, 8, and 15 Are Patentable Over <u>*Hetherington*</u> and <u>*Kaufman*</u>

Applicants respectfully submit that the Examiner has omitted one or more essential elements needed for a *prima facie* rejection. The cited art, alone or in combination, fails to teach or suggest every limitation of the claims. For example, the combination of *Hetherington* and *Kaufman* fails to teach or suggest at least "mapping a user-specific process so that it overlays virtual addresses of the master process," as recited in claim 1.

1. "Forking" Is Not The Same As Overlaying Virtual Addresses

The Examiner asserts that "Kaufman teaches that when spawning, the child would overlay the parent's virtual memory address (vm_folk (sic) to duplicate a parent process's virtual memory information for a child process, Col. 31 line 14-20 and Col. 2 lines 1-5)." (*See, e.g.* the Advisory Action). Applicants respectfully submit that, as disclosed in *Kaufman*, a child process is formed by copying but does not share or overlay the virtual addresses of the parent. Information of the parent may be copied in *Kaufman*, but the virtual address is not overlaid or the same virtual address, for example. Copying or duplication of information is not the same as sharing the same virtual addresses, and *Kaufman* equates forking with copying or duplication. For example, *Kaufman* states "[a] fork element can create a second process that initially duplicates a first one and can initiate generation of a new-process signal in connection with creation of the second process." (Abstract of *Kaufman*; emphasis added). Nowhere does it teach or suggest that the processes share or overlay the same virtual addresses. In addition, *Kaufman* states "many computer systems, for example, those running under UNIX or UNIX-like operating systems, permit process duplication, or forking. Forking causes one process to replicate, spawning a new process. The first process, referred to as the parent, continues in the normal manner after the fork. The spawned process, or child, though initially identical to the parent, can be executed in a different manner." (Col. 2, ll. 1-5 of *Kaufman*; emphasis added). In the portion cited by the Examiner, *Kaufman* states "[t]he vm system executes the steps of the procedure vm_fork to duplicate a parent process's virtual memory information for a child process." (Col. 31, ll. 14-21 of *Kaufman*). As stated above, duplication of information is not sharing or overlaying the same virtual addresses.

2. Context Address Space Mapping Is Not Virtual Address Space Mapping

In response to Applicants' arguments that the duplication of information, or "forking," disclosed in *Kaufman* is not the same as overlaying the same virtual address, the Examiner

further asserted that "Kaufman teaches the mapping so that the child overlaid the virtual address of the parent (vm_fork, syscall_finish_fork, vm_mapin to mapin [sic] the overlay object and mapping out a file range from a process's context address space, col. 31 lines 15-65 and col. 34 lines 10-15)" (*See* Final Official Action of 4/19/2005, p. 8, no. 24). However, the Examiner's argument with regard to mapping in and out of a process's context address space still fails to establish *prima facie* obviousness.

The portion of *Kaufman* cited by the Examiner states that "[t]he VM system executes the steps of MAPOUT as a user entry for mapping out a file range from a process's <u>context address</u> space. (*See* Col. 34, ll. 10-15 of *Kaufman*). The procedure accepts as input a handle of the map, and returns a status of the mapping procedure" (Emphasis added). However, claim 1 recites a <u>virtual</u> address overlay and not a <u>context</u> address overlay or mapping. A context address is very different from a virtual address, as explained in *Kaufman* itself:

> The memory architecture of system 10 consists of two levels of related address space: context address (CA) space and system virtual address (SVA) space. Context address space is the programmer's interface to memory. There are many context address spaces in a system. System virtual address space stores the data from all context address spaces. There is only one system virtual address space. Another address space, the system physical address space (SPA) defines hardware control registers. (Col. 18, ll. 29-38)

Thus, the MAPIN and MAPOUT features of *Kaufman* merely allow forked processes to map to a section of context address space of a parent process, and do not provide mapping to the virtual address of a parent process. (*See* Col. 30, ll. 4-14 and Col. 33, ll. 25-30 of *Kaufman*).

In the Advisory Action mailed September 13, 2005, the Examiner contended that the virtual address space stores all data from the context address space and that each segment of the context address space is mapped to the system virtual address space. The Examiner further asserted that when a process's context address space is mapped, the process's virtual address

space is also mapped. That assertion is false for at least two reasons. Firstly, *Kaufman* does not at all teach or suggest that the virtual address space is mapped out with the context address space. *Kaufman* merely discloses that the VM system executes the steps of MAPOUT as a user entry for mapping out a file range from a process's context address space (*See* Col. 34, ll. 10-15 of *Kaufman*). Secondly, there is no per-process virtual address space in *Kaufman*. *Kaufman* explicitly states that there is only one virtual address space. (*See* Col. 18, ll. 35-36 of *Kaufman*).

As a result, Applicants submit that claim 1 is patentable for at least the above-mentioned reasons. Claims 3-7 depend on claim 1 and are therefore patentable at least for the same reasons. Claim 15 is patentable for at least the same reasons as claim 1. Furthermore, claims 17-21 depend on claim 15 and are therefore patentable at least for the same reasons.

Applicants further submit that claims 8 and 12 are also patentable for at least the same reasons as claim 1. Just as *Kaufman* fails to teach or suggest "mapping a user-specific process so that it overlays virtual addresses of the master process," *Kaufman* also fails to teach or suggest mapping "a user-specific process to virtual addresses that mirror virtual addresses of the global process." Furthermore, claims 9-11 and 13-14 depend on claims 8 and 12 respectively and are therefore patentable at least for the same reasons.

B. Claims 2 and 16 Are Patentable Over *Hetherington* and *Kaufman*

Applicants respectfully submit that the Examiner has omitted one or more essential elements needed for a *prima facie* rejection. The cited art, alone or in combination, fails to teach or suggest every limitation of the claims. For example, the combination of *Hetherington* and *Kaufman* fails to teach or suggest at least "transferring data between the master process and the user-specific process using a communications channel that does not require the serialization of data," as recited in claim 2. The Examiner asserts that the interprocess communication (IPC) facility of *Hetherington* teaches this limitation. (*See* Col. 4, ll. 13-19 of *Hetherington*).

However, IPC, including remote procedure call (RPC), typically employs data serialization. (*See* page 2, ll. 22-27 of the patent application). Data is serialized at a first end, transferred between processes, and then deserialized at the other end. *Hetherington* fails to disclose that its IPC facility does not employ data serialization, and therefore teaches away from the limitations of claim 2. Claim 16 recites similar limitations, and is therefore patentable for at least the same reasons as given for claim 2.

IX. Conclusion

Applicants respectfully submit that the outstanding rejections should be reversed, and that

the application is in condition for allowance.

Respectfully submitted,

Dated: December 28, 2005   By:_____

A. Wesley Ferrebee, Reg. No. 51,312

Customer No. 58328 (formerly 26263)
SONNENSCHEIN NATH & ROSENTHAL LLP
P.O. Box 061080
8000 Sears Tower
Chicago, IL 60606-6404
Telephone: (202) 408-6832
Facsimile: (312) 876-7457

APPENDIX A: Claims on Appeal

1. (Original) In a computer system, a method for providing for concurrent subprocessing of a master process, the method comprising the steps of:

interfacing with a master process when a user-specific operation is encountered;

mapping a user-specific process so that it overlays virtual addresses of the master process; and

processing the user-specific operation in the user-specific process.

2. (Original) The method of claim 1, further comprising the step of:

transferring data between the master process and the user-specific process using a communications channel that does not require the serialization of data.

3. (Original) The method of claim 1, further comprising the step of:

providing an interface for the user-specific process that mirrors an interface for the master process.

4. (Previously presented) The method of claim 1 wherein the master process is a global locale process and the user-specific process is a locale-specific process.

5. (Original) The method of claim 1 wherein the user-specific process is mapped after the user-specific operation is encountered.

6. (Original) The method of claim 1 wherein the user-specific process is mapped before the user-specific operation is encountered.

7. (Original) The method of claim 1 further comprising the step of:

returning processing to the master process after processing the user-specific operation in the user-specific process.

8. (Previously presented) A computer-readable medium comprising computer instructions that facilitate concurrent handling of subprocesses in a system that utilizes a global process, wherein the instructions, when executed, cause the system to perform the steps of:

interfacing with the global process when a user-specific operation is encountered;

mapping a plurality of concurrent user-specific processes, wherein each user-specific process is mapped to virtual addresses that are equivalent to virtual addresses of the global process; and

processing the user-specific operation in one of the user-specific processes.

9. (Previously presented) The computer-readable medium of claim 8, wherein the instructions, when executed, provide each of the plurality of concurrent user-specific processes with an interface that is identical to an interface of the global process.

10. (Previously presented) The computer-readable medium of claim 9, wherein the instructions, when executed, cause the system to perform the step of mapping subprocesses within each of the plurality of user-specific processes, the subprocesses being mapped to virtual addresses that are equivalent to virtual addresses for user-specific operations of the global process.

11. (Previously presented) The computer-readable medium of claim 10, wherein the instructions, when executed, cause the system to perform the step of returning processing to the global process after execution of the subprocesses is complete.

12. (Previously presented) A computer system for enabling concurrent multiple subprocess handling in a global process environment, the system comprising:

a global process; and

a virtual memory separator that maps a user-specific process to virtual addresses that mirror virtual addresses of the global process, the user-specific process having an interface that mirrors an interface of the global process.

13. (Previously presented) The computer system of claim 12 wherein the global process is a global locale process and wherein the user-specific process is a locale-specific process.

14. (Previously presented) The computer system of claim 12 wherein the global process is a global daemon process and wherein the user-specific process is a user-specific daemon process.

15. (Original) An apparatus for conducting multi-user concurrent handling of subprocesses, the apparatus comprising:

means for interfacing with a master process when a user-specific operation is encountered;

means for mapping a user-specific process so that it overlays virtual addresses of the master process; and

means for processing the user-specific operation in the user-specific process.

16. (Original) The apparatus of claim 15, further comprising:

means for transferring data between the master process and the user-specific process using a communications channel that does not require the serialization of data.

17. (Original) The apparatus of claim 15, further comprising:

means providing an interface for the user-specific process that mirrors an interface for the master process.

18. (Previously presented) The apparatus of claim 15 wherein the master process is a global locale process and the user-specific process is a locale-specific process.

19. (Original) The apparatus of claim 15 wherein the user-specific process is mapped after the user-specific operation is encountered.

20. (Original) The apparatus of claim 15 wherein the user-specific process is mapped before the user-specific operation is encountered.

21. (Original) The apparatus of claim 15, further comprising:

means for returning processing to the master process after the user-specific operation is executed in the user-specific process.

APPENDIX B: Evidence

None.

APPENDIX C: Related Proceedings

None.